520.43578X00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants: Y. AKITA

Serial No.:

Filed: March 3, 2004

For: Logic Circuit And Program For Executing Thereon

Group:

Examiner:

LETTER CLAIMING RIGHT OF PRIORITY

Mail Stop: New Appln.
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

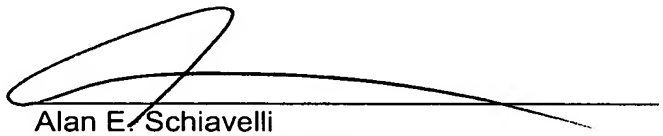
March 3, 2004

Sir:

Under the provisions of 35 USC 119, applicant hereby claims the right of priority based on Japanese Patent Application No. 2003-128086, filed May 6, 2003.

Respectfully submitted,

ANTONELLI, TERRY, STOUT & KRAUS, LLP



Alan E. Schiavelli
Registration No. 32,087

AES/jla
(703) 312-6600
Attachment

日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日 Date of Application:	２００３年　５月　６日
出願番号 Application Number: [ST. 10/C]:	特願２００３－１２８０８６ ［ＪＰ２００３－１２８０８６］
出願人 Applicant(s):	株式会社日立製作所

２００４年　２月１２日

特許庁長官
Commissioner,
Japan Patent Office　今井康夫

出証番号　出証特２００４－３００８６５０

【書類名】　特許願

【整理番号】　NT03P0235

【提出日】　平成15年 5月 6日

【あて先】　特許庁長官　殿

【国際特許分類】　G06N　3/04

【発明者】

【住所又は居所】　東京都国分寺市東恋ヶ窪一丁目２８０番地　株式会社日立製作所　中央研究所内

【氏名】　秋田　庸平

【特許出願人】

【識別番号】　000005108

【氏名又は名称】　株式会社日立製作所

【代理人】

【識別番号】　100068504

【弁理士】

【氏名又は名称】　小川　勝男

【電話番号】　03-3661-0071

【選任した代理人】

【識別番号】　100086656

【弁理士】

【氏名又は名称】　田中　恭助

【電話番号】　03-3661-0071

【選任した代理人】

【識別番号】　100094352

【弁理士】

【氏名又は名称】　佐々木　孝

【電話番号】　03-3661-0071

【手数料の表示】

【予納台帳番号】　081423

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　論理回路及びその論理回路上で実行するプログラム

【特許請求の範囲】

【請求項１】

論理演算又は算術演算を行う演算回路と、前記演算回路を制御する制御回路とを具備し、

前記制御回路は、演算回路に対して実行すべき演算の種別を規定する複数の命令と前記複数の命令の間の依存関係を示す情報とを含むプログラムが入力され、前記プログラムに従って前記演算回路を制御することを特徴とする論理回路。

【請求項２】

請求項１において、

前記制御回路は、前記依存関係を示す情報に従って、前記複数の命令の実行順序を決定し、前記演算回路に前記複数の命令のうち実行可能な命令を供給することを特徴とする論理回路。

【請求項３】

請求項２において、

前記依存関係を示す情報は、前記複数の命令のうち対応する命令を実行するために、既に実行が終了していなくてはいけない先行命令の情報であり、

前記制御回路は、前記先行命令が実行されたか判断することを特徴とする論理回路。

【請求項４】

請求項２において、

前記論理回路は、前記演算回路を複数有し、

前記制御回路は、前記複数の命令のうち実行可能な命令を並列に前記演算回路に出力することを特徴とする論理回路。

【請求項５】

請求項１において、

前記論理回路は、リコンフィギャラブルプロセッサであり、

前記演算回路は、複数の演算種を含むと共にアレイ状に配置され、

前記プログラムは、演算の入出力として使用するデータの定義と、前記演算回路に対する前記演算種の指定と、前記アレイ状に配置された演算回路間の配線の接続状態の指定と、前記アレイ状に配置された演算回路のうち対応する演算回路が演算を行うために必要な入力データの情報とを含み、

前記制御回路は、入力された前記プログラムに従って、前記アレイ状に配置された演算回路間の配線の接続状態を制御し、対応する前記演算回路が実行可能かを判断することを特徴とする論理回路。

【請求項６】

論理演算や算術演算などを行う演算回路と、演算回路を制御する制御回路とを有する論理回路に対して、前記制御回路を介して前記演算回路に指示を与えて目的の演算を前記論理回路に実行させるプログラムであって、

前記演算回路に対して実行すべき演算の種別を規定する命令、または複数の演算回路に対して実行すべき演算群の種別を規定する命令群を含み、かつ、前記命令又は前記命令群の間に存在する実行順序の依存関係が記述されていることを特徴とするプログラム。

【請求項７】

請求項６に記載のプログラムにおいて、

前記複数の命令または前記命令群から構成される命令ブロックを定義し、前記命令ブロック間の実行順序の依存関係が記述されていることを特徴とするプログラム。

【請求項８】

請求項６または７に記載のプログラムにおいて、

前記命令、前記命令群、または前記命令ブロックの間に存在する実行順序の依存関係を、

前記命令、前記命令群、または前記命令ブロックからなる演算群と、

前記命令、前記命令群、または前記命令ブロックの入力、または出力となるデータと、

前記演算群と、前記演算群の実行に必要とするデータとの間の関係と、

前記演算群と、前記演算群が生成するデータとの間の関係と、

が記述されていることを特徴とするプログラム。

【請求項９】

請求項６に記載のプログラムにおいて、

前記命令または前記命令群が規定する演算を開始するために、実行が終了していなくてはならない先行命令が記述されていることを特徴とするプログラム。

【請求項１０】

請求項６から９のいずれかに記載のプログラムは、前記演算回路がアレイ状に配置され、前記演算回路に対する演算種の指定と、前記演算回路間の接続を指定することにより動作を制御するリコンフィギャラブルプロセッサを対象とすることを特徴とするプログラム。

【請求項１１】

請求項１０に記載のプログラムおいて、

演算の入出力として使用するデータの定義と、

前記演算回路に対する演算種の指定と、

前記演算回路間の配線の指定とを、一つまたは複数の演算回路に対して指定することにより定義した命令ブロックが、演算を行うために必要な入力データの情報を有することを特徴とするプログラム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、論理回路と、論理回路上で実行するプログラムに関する。

【０００２】

【従来の技術】

マイクロプロセッサの性能は年々向上してきている。性能向上の要因には、製造技術やアーキテクチャの改善が挙げられ、今後もこれらの技術の革新により、さらなる性能向上が期待されている。

【０００３】

アーキテクチャ改善による性能向上の一例に、スーパースカラやVLIW（Very Long Instruction Word）の採用がある。いずれも、複数の演算回路をハードウェ

アとして実装することにより、複数の命令を同時に実行し、プロセッサの性能を向上させる技術である。

【０００４】

スーパースカラ、VLIWの両者は、複数命令を実行することにより処理性能を向上させる点で共通している。通常、プロセッサの実行時には、どのような演算を行うかを記述したプログラム（オブジェクトコード）が与えられる。スーパースカラやVLIW以前のプロセッサでは、プログラムを一個ずつ逐次実行することを前提とした命令列により与えており、命令列は先頭から一個ずつ逐次実行すれば正しい演算結果が得られることは、プログラムの作成者により保障されていた。

【０００５】

しかし、プログラム中の複数命令を同時に実行した場合、正しい結果が得られる場合と、得られない場合がある。これは命令の実行順序に依存関係がある場合とない場合が存在するからであり、任意に選択した複数命令を同時に実行した場合は、一般には正しい結果を得ることができない。そのため、スーパースカラやVLIWでは、命令間の依存関係の解析を行い、正しい結果が得られる場合のみ、複数命令の同時実行を行っている。依存関係解析の手法は、以下のように両者で異なっている。

【０００６】

スーパースカラは、命令間の依存関係を評価し、同時実行可能な命令を検出する機構をハードウェアとして備える。スーパースカラを採用したプロセッサ（以下、「スーパースカラプロセッサ」と呼ぶ）はそれ以前のプロセッサと同様に、一命令ずつ実行することを前提としたプログラムを入力とするが、プログラム実行時にハードウェアで命令間の依存関係を調査し、正しい結果が得られると判断できた場合のみ、複数の命令を同時に実行する。

【０００７】

スーパースカラは、複数のプロセッサでプログラムを共有できることが利点である。つまり、スーパースカラプロセッサでは、プログラム中に命令の依存関係に関する情報を持っていないため、スーパースカラ以前のプロセッサや、同時実行可能な命令数の異なるスーパースカラプロセッサ間で、同一のプログラムを実

行することができる。従って、同一のプログラムを用いても、プログラム実行時にハードウェアで依存関係の調査を行うため、同時実行可能な命令数の多いプロセッサでは、より多くの命令を同時に処理し、高い性能を出すことができる。このようなスーパースカラプロセッサについては、例えば、非特許文献１に記載されている。

【０００８】

これに対しVLIWは、プログラム作成時に命令間の依存関係をあらかじめ調査しておく。通常、プロセッサ向けのプログラムを作成する場合にはコンパイラを使用するが、VLIWを採用したプロセッサ（以下、「VLIWプロセッサ」と呼ぶ）向けのコンパイラは、コード生成時に命令間の依存関係の評価も同時に行う。また、VLIWプロセッサ向けのプログラム（オブジェクトコード）は、同時に実行すべき命令を明示的に示す構造となっている。コンパイラは依存関係の評価結果をもとに、スケジューリング（同時実行する命令の組み合わせの決定）を行い、その結果をオブジェクトコード内に記述する。この方式は、命令間依存関係の調査をハードウェアで行う必要がないため、ハードウェア量が少なくてすむことが利点である。このようなVLIWプロセッサについては、例えば、非特許文献２に記載されている。

【０００９】

また、高い演算性能とフレキシビリティを同時に実現するLSI（Large Scale Integrated Circuit）として、リコンフィギャラブル（Re-configurable Processor）プロセッサが近年注目されている。リコンフィギャラブルプロセッサは、アレイ状に配置されたALU（Arithmetic Logic Unit）などの演算回路と、演算回路間を接続するスイッチから構成される。演算回路の機能と、演算回路間の配線は、コンフィグレーションレジスタと呼ばれるレジスタの内容により再構成することが可能であり、目的に応じて構成内容を変えることにより、プログラムを実行する。リコンフィギャラブルプロセッサの中でも、プログラムの実行中にコンフィグレーションレジスタの内容を変更できるものがダイナミックリコンフィギャラブルプロセッサと呼ばれ、近年特に注目されている。

【００１０】

リコンフィギャラブルプロセッサの演算回路は、ALUが持つ加減算やNANDやNORなどの論理演算など、複数の演算が実行可能であり、それらのうちどの機能を選択するかは、コンフィグレーションレジスタの内容により決定される。また、演算の入力信号をどこから得るか、あるいは演算の出力をどこに出力するかなどは、スイッチの接続により決まり、スイッチの接続もコンフィグレーションレジスタの内容により決定される。リコンフィギャラブルプロセッサに対するプログラムは、このコンフィグレーションレジスタに対する設定を与えるものである。

【００１１】

リコンフィギャラブルプロセッサには、アレイを大きくすることにより性能向上が可能であるという特徴がある。つまり、半導体の製造技術の進歩などにより、チップ上に集積可能なトランジスタ数が増加した場合、演算回路数を増加させ、アレイを大きくすることにより、同時に実行可能な演算数を増やして性能を向上させることが可能であり、性能スケーラビリティがよい。なお、ここで、「性能スケーラビリティ」とは使用可能なトランジスタ数が増加した場合に、トランジスタ数に比例して性能を向上させられることを言う。このようなリコンフィギャラブルプロセッサについては、例えば、非特許文献３に記載されている。

【００１２】

【非特許文献１】

Sohi, G. S, “Instruction issue logic for high-performance, interruptible, multiple functional unit, pipelined computers”, IEEE Transactions on Computers, Vol. 39, No. 3, March 1990, pp. 349-359.

【非特許文献２】

Fisher, J. A, “Very Long Instruction Word Architecture and the ELI-512”, Proceedings of the 10th International Symposium on Computer Architecture, 1983.

【非特許文献３】

R. Hartenstein, “Coarse Grain Reconfigurable Architectures”, ASP-DAC 2001, pp. 564-569.

【００１３】

【発明が解決しようとする課題】

しかしながら、前述したようにプロセッサのプログラム実行方式として、スーパースカラ、VLIWの二つがあるが、それぞれハードウェア量、プログラムの互換性に関して欠点がある。つまり、スーパースカラでは、命令間の依存関係をハードウェアで評価するため、性能の異なるプロセッサ間でプログラムの互換性があるという利点があるものの、依存関係を調査する専用のハードウェアを持つため、ハードウェア量が増大するという欠点がある。

【００１４】

一方VLIWでは、コンパイラであらかじめ命令間の依存関係を調査し、スケジューリングを行っておくため、LSI上のハードウェア量は少なくて済むという利点があるものの、コンパイルの段階でスケジューリングを行うため、プログラム（オブジェクトコード）を複数種類のプロセッサ間で共有できないという欠点がある。つまり、コンパイラがプロセッサの持つ演算回路数を考慮したスケジューリングを行うため、別のVLIWプロセッサ向けにコンパイルされたオブジェクトコードは、異なる演算回路数のVLIWプロセッサでは使用できなくなり、プロセッサ間でのプログラム互換性がない。

【００１５】

したがって、スーパースカラ、VLIWの両方式では、少ないハードウェア量で、しかも異なるプロセッサ間でプログラムの互換性を保つということができない。

【００１６】

また、現在利用されているリコンフィギャラブルプロセッサに対するプログラムは、特定の大きさのアレイに対するプログラムであり、異なるアレイサイズを持つリコンフィギャラブルプロセッサではそのプログラムを実行することができないという欠点がある。

【００１７】

そこで、本発明の目的は、異なるハードウェアで互換性を保ち、高い演算性能を実現できると共に、ハードウェア量を削減できる記述形式のプログラムを提供することにある。

【００１８】

また、そのプログラムを読み込み実行するのに最適な論理回路、及びプロセッサを提供することも本発明の目的の一つである。

【００１９】

【課題を解決するための手段】

本発明に係るプログラム及び論理回路の代表的手段の一例を示せば次の通りである。

【００２０】

本発明に係るプログラムは、論理演算や算術演算などを行う演算回路と、前記演算回路を制御する制御回路とから構成される論理回路に対して、前記制御回路を介して演算回路に指示を与えることにより、目的の演算を論理回路に実行させるプログラムであって、前記演算回路に対して実行すべき演算の種別を規定する命令、または複数の演算回路に対して実行すべき演算群の種別を規定する命令群を含み、かつ、前記命令又は命令群の間に存在する、実行順序の依存関係が記述されていることを特徴とするものである。

【００２１】

また、本発明に係る論理回路は、論理演算又は算術演算を行う演算回路と、前記演算回路を制御する制御回路とを具備し、前記制御回路は、演算回路に対して実行すべき演算の種別を規定する複数の命令と前記複数の命令の間の依存関係を示す情報とを含むプログラムが入力され、前記プログラムに従って前記演算回路を制御することを特徴とする。

【００２２】

【発明の実施の形態】

以下、本発明の好適な実施形態について、具体的な実施例を用いて添付図面を参照しながら詳細に説明する。

【００２３】

＜実施例１＞

本発明のプログラム、及びそのプログラムを実行する論理回路の一実施例を示す。

【００２４】

本実施例は、図１に示すように、実行する演算OP1～OP5からなる演算群と、演算に要するデータ依存関係、すなわち演算の実行順序の制約109（図中に、制約があることを小さい丸を付加した矢印で示す）とを含むプログラム（PRG）100と、そのプログラムを実行する論理回路LGCから構成される。ここでは一例として、論理回路LGCは、一つの制御回路CTRと、３つの演算回路ALU1～ALU3で構成される。

【００２５】

プログラム100には、論理回路LGCで実施すべき演算OP1を記述するほか、演算で使用するデータの受け渡しに起因する演算の実行順序の制約109を記述する。プログラム100内に書かれる演算は、演算の実行順序制約109が規定する実行順序制約を満たしていれば、どの順序で演算を行っても正しい結果が得られることをプログラム作成者により保障されている。

【００２６】

このプログラム100を読み込み実行する論理回路LGCは、内部に３つの演算回路ALU1～ALU3を持ち、同時に３個の演算を実行可能である。そのため、演算回路ALU1～ALU3を制御する制御回路CTRは、論理回路LGCがプログラム全体を短時間で終了させるために、最大３個の並列実行可能な演算を抽出し、同時に実行するように演算回路ALU1～ALU3に指示を出す。この例では、５個ある演算OP1～OP5のうち、演算OP3及び演算OP4は、それぞれ演算OP1、OP2及びOP5の終了後でなくては実行できないが、演算OP1、演算OP2、及び演算OP5は並列に実行しても実行順序制約に違反しないため、同時実行が可能である。そのため制御回路CTRは、演算回路ALU1～ALU3に対し、まず演算OP1、OP2、OP5をそれぞれ実行させ、その後、演算OP3、及び演算OP4を実行させることにより、２ステップでプログラム全体の実行を完了する。

【００２７】

図２は、本実施例のプログラム100と、制御回路CTRを、より詳細化した図面である。図２では、図１でのプログラム100と同一の内容を表現しているが、プログラム100内部で表現する実行順序制約109を、演算で使用するデータを用いて表している。つまり、演算を示すOP1などだけでなく、演算の入出力となるデータ

として、入力データ123（In-Data1～In-Data3）、出力データ122（Out-data1、Out-data2）、およびデータ（DATA1～DATA3）を用い、これらデータと演算との間の関係をデータフローグラフとして表現することにより、実行順序を規定している。

【００２８】

具体的には、演算OP1は、入力データ123の一部であるIN-Data1を用いて演算を行うが、入力データはプログラム100の実行時には必ず用意されているため、演算OP1は任意の時刻で実行可能な演算となる。演算OP1は、実行を終えると演算結果としてDATA1を生成する。演算OP2、演算OP5についても同様であり、それぞれDATA2、DATA3を生成する。

【００２９】

演算OP3は、演算の入力としてDATA1を使用する。入力データ123と異なり、プログラムの内部データであるDATA1は、プログラム実行開始当初には用意されておらず、利用不可である。DATA1が利用可能になるのは、このデータを生成する演算OP1が実行を終えた後であり、このため演算OP3は演算OP1の実行後にしか実行できないという制約が生じる。演算OP4も演算OP3と同様であり、演算OP4の実行にはDATA2、DATA3が必要であることから、演算OP2及び演算OP5の実行後にしか演算OP4が実行できないという制約が生じる。

【００３０】

図２の論理回路LGC中の制御回路CTRは、前記のプログラム100を読み込み、実行すべき演算を選択する機構を詳細化したものである。制御回路CTRは、演算管理部OM、データ管理部DM、実行演算選択部OSの３つから構成される。また、図９には実行演算選択部OSの処理内容の概略を示した。

【００３１】

プログラム実行開始前に、制御回路CTRがプログラム100を読み込み、演算とデータとに分離して、それぞれ演算管理部OM、データ管理部DMに格納する。演算管理部OMへの格納時には、図１１に示すように演算名（OP1,OP2,OP3,…等）とその演算で必要とする入力データ名（In-Data1,In-Data2,DATA1,…等）、データ管理部DMへの格納時には、図１２に示すように演算名とその演算で必要なデータ名が

格納される。データ名が格納されていれば利用可であり、データ名が格納されていなければ未だそのデータが利用不可の状態である。図１２では、一例として、プログラム実行開始時の状態、すなわちまだ演算OP3とOP4で必要なデータ名DATA1、DATA2、DATA3が格納されていない状態を示している。データの利用可・不可は、プログラム実行開始時には、使用可能な入力データのみ利用可となり、その他のデータは利用不可となる。プログラムの実行が進み、新たにデータが生成されると、その段階でデータ管理部DMに利用可能としてデータ名が格納される。なお、利用可、利用不可については、データ名の外に利用可、利用不可のビットを設けて判断できるようにしてもよい。

【００３２】

プログラムを実行する際には、実行演算選択部OSが、演算管理部OMから演算名（OP）を取得する（図９のステップS90）。次に、演算OPの入力データの状態をデータ管理部DMから取得する（ステップS91）。これら取得した演算管理部OMとデータ管理部DMからの情報をもとに、演算が実行可能か（すなわち、演算に必要なデータが利用可能か）を判定し、実行すべき演算を決定する。演算が実行可能であるかの判定は、演算管理部OMから受け取る演算実行に必要なデータに関する情報と、データ管理部DMから受け取る必要なデータが利用可能であるかの情報とを組み合わせて行い、演算実行に必要な全てのデータがそろったものを実行可能であると判定する。

【００３３】

判定の結果、実行不可であればステップS90に戻って次の演算OPを取得し、実行可能であれば、同時実行可能な演算数以下、つまり演算回路ALUの個数以下、本実施例では演算回路の数はALU1～ALU3の３個であるから、３個以下の演算が同時に実行可能であると判定された場合は、それらの演算を全て同時に実行するように各演算回路に対して指示を出す（ステップS92）。同時実行可能な演算数が演算回路の個数より大きい場合には、同時実行可能な演算から演算回路の個数と等しい演算を、先に演算管理部OMに入れられたものから選び、実行する。次に、実行後の演算OPが生成したデータを利用可能に修正、すなわちデータ名をデータ管理部DMに格納する（ステップS93）。

【００３４】

本実施例によれば、論理回路に与えるプログラム内に実行すべき演算と、その演算を実行するための実行順序制約（依存関係）を記述し、プログラムを実行する論理回路は制御回路により、読み込んだプログラムに記述された実行順序制約に基いて演算回路の実行順序を決定して演算を実行する。これにより、異なる性能のハードウェア上での互換性を保ちつつ、高い性能スケーラビリティを実現できる。

【００３５】

＜実施例２＞

本発明のプログラム及びそのプログラムを実行するプロセッサの一実施例を示す。本実施例は図３に示すように、プログラム200と、それを実行するプロセッサ204とから構成される。また、図１０にはディスパッチャ210の処理内容の概略を示した。

【００３６】

プログラム200は、複数の命令INST1，INST2，INST3，INST4，…からなり、命令には、実行順序を規定する制約に関する情報を持つ。制約として持つ情報は、命令間に実行順序制約が存在する場合、先行して実行すべき命令には、先行命令であることを示す情報、先行命令の実行完了後に実行すべき命令は、実行が完了していなくてはならない先行命令のアドレスである。図３では一例として、命令INST1と命令INST3の命令間と、命令INST2と命令INST4の命令間に実行順序制約209（図中に、小さい丸を付加した矢印で示す）がある場合を示している。

【００３７】

プロセッサ204は、制御回路CTRと演算回路を持つ。制御回路CTRは、プログラム200のフェッチ、デコードなどを含み、プログラム中の命令を演算回路に振り分けるディスパッチャDPTと、実行順序の制御に使用する実行済み命令リストEILから構成される。演算回路は、ここでは一例として３個の演算回路ALU1～ALU3で構成され、それぞれの演算回路が同時に異なる命令を実行可能である。

【００３８】

プログラムの実行時には、ディスパッチャDPTがプログラム200を読み込み、プ

ログラムから命令を取得する（図１０のステップS10）。取得した命令の先行命令の実行状態を、実行済み命令リストEILから取得し（ステップS11）、先行命令が未実行であればステップS10にもどり、実行済みであれば、次のステップS12に進む。

【００３９】

ステップS11での個々の命令が実行可能であるかの判定は、実行順序制約を用いて行う。もし、判定対象の命令に対する実行順序制約が存在しない場合、その命令は実行可能であるとする。実行順序制約があり、完了していなくてはならない先行命令が存在する場合、そのアドレスが実行済み命令リストEILに存在されているかを確認し、存在する場合は実行可能、存在しない場合は実行不可能と判定する。

【００４０】

ディスパッチャDPTは、先頭から順に実行可能な命令を実行するように演算回路ALUに対し指示を出す（ステップS12）が、実行を完了した命令が実行順序制約における先行命令になっていた場合、その命令のアドレスを実行済み命令リストEILに追加記録する（ステップS13）。

【００４１】

なお、プログラムの分岐命令を実行した後には、実行済み命令リストEILを初期化する。

【００４２】

本実施例によれば、プロセッサに与えるプログラム内に実行すべき命令と、その命令を実行するための実行順序制約（依存関係）を記述し、プログラムを実行するハードウェアは制御回路内のディスパッチャDPTにより、読み込んだプログラムに記述された実行順序制約に基いて演算回路への命令の割り当てと実行順序を決定して、実行する。これにより、異なる性能のプロセッサ上でのプログラム互換性を保ちつつ、高い性能スケーラビリティを実現できる。

【００４３】

＜実施例３＞

本発明のプログラム及びそのプログラムを実行するリコンフィギャラブルプロ

セッサの一実施例を示す。図４は、リコンフィギャラブルプロセッサを構成する演算回路アレイである。このリコンフィギャラブルプロセッサは、４×４個の演算回路セルALUCにより構成さる。演算回路アレイ300は、データ転送用のデータバス302と、コンフィグレーションデータ転送用のコンフィグレーションバス303を持つ。演算回路セルALUCはデータバス302を通じて、メモリ、他の演算回路アレイ、他のモジュール、あるいは他のチップに接続される。またコンフィグレーションデータは、コンフィグレーションバス303を通じてコンフィグレーションメモリに書き込まれる。

【００４４】

図５は、図４の各演算回路セルALUCの内部構造を示す図である。演算回路セルALUCは、コンフィグレーションメモリCFG_MEMおよび選択回路SELと、加算回路（ADD）403、NAND回路404、NOR回路405、……等の複数の異なる機能の各種回路とを含んでいる。通常、リコンフィギャラブルプロセッサのアレイ300を構成する各演算回路セルALUCは、それぞれ上記のような複数の異なる機能の回路を持ち、目的の演算に応じて使用する回路を切り替える。どの回路を選択するかは、コンフィグレーションメモリCFG_MEMが記憶し、その内容に応じて選択回路SELが、回路403、404、405…の中から、必要とする機能の回路の入出力を選択する。

【００４５】

コンフィグレーションメモリCFG_MEMの内容は、コンフィグレーションバス303を通じて、外部からコンフィグレーションメモリCFG_MEMに書き込まれる。各回路403～405等は、選択回路SELによりどれか一つが選択され、演算を行う。選択された回路は、演算回路セルALUCのデータバス302の入力ポートINからデータが選択回路SELを通じて入力され、演算を行い、その結果を選択回路SELを通じて演算回路セルALUCのデータバス302の出力ポートOUTに出力する。

【００４６】

図６は、リコンフィギャラブルプロセッサの全体像を示す図である。このリコンフィギャラブルプロセッサ500は、複数の演算回路アレイ300と、演算回路アレイ間を続する接続素子501、メモリMEM、およびコンフィグレーション制御回路CFG_CTRから構成される。個々の演算回路アレイ300は、図４に示したように演算回

路セルALUCから構成され、図５に示したようにコンフィグレーションメモリCFG_MEMの内容を書き換えることにより、様々な演算を行うことができる。

【００４７】

演算に要する入出力データは、データバス302と接続素子501とを経由して、メモリMEMや他の演算回路アレイ300の出力、あるいはプロセッサ外から受け取る。接続素子501は、演算回路アレイ300間の接続を行う素子であり、外部からの指示に応じて演算回路アレイ間の接続、他モジュール、メモリ、あるいはチップ外部などと接続する。リコンフィギャラブルプロセッサ500は、プロセッサ全体で処理する演算を分割し、内部に持つリコンフィギャラブルアレイすなわち演算回路アレイ300に対し分配することにより、処理を行う。

【００４８】

演算回路アレイ300の入出力データを格納するために要するメモリMEMに対しても、接続素子501を経由してアクセスする。また、個々の演算回路アレイ300に対するコンフィグレーションデータの書き込みは、コンフィグレーション制御回路CFG_CTRが行い、コンフィグレーションバス303を経由してコンフィグレーションデータを書き込む。

【００４９】

図７は、リコンフィギャラブルプロセッサ500へ与えるプログラムの構造を示したものである。プログラム600は、内部に演算回路アレイ300に対するプログラムALU-ARRAY_PRG1、ALU-ARRAY_PRG2、ALU-ARRAY_PRG3、……を持つ。

【００５０】

図８は、演算回路アレイ300に対するプログラムALU-ARRAY_PRG1の構造を示したものである。プログラムALU-ARRAY_PRG1は、入力データIn-data、出力データOut-data、及び各演算回路セルALUCに対するプログラムALUC_PRG1-1、ALUC_PRG1-2、……から構成される。

【００５１】

入力データIn-dataは、演算回路アレイ上でプログラムALU-ARRAY_PRG1を実行するために必要な入力データを示したものであり、プログラム600全体の中で、サブプログラム（演算回路アレイへのプログラム）の実行順序を規定する制約と

なる。出力データOut-dataは、演算回路アレイが出力するデータを示すものである。ある演算回路アレイが実行を完了すると、その演算回路アレイが出力したデータが、他のアレイで入力として使用可能になる。

【００５２】

各演算回路セルに対するプログラムALUC_PRG1-1、ALUC-PRG_1-2、……は、演算回路アレイ内に含まれる個々の演算回路セルALUCに対してのプログラムであり、演算回路セルALUC内に含まれるコンフィグレーションメモリCFG-MEMの内容を示すものである。

【００５３】

リコンフィギャラブルプロセッサ500全体の管理は、コンフィグレーション制御回路CFG_CTRが行う。この回路がプログラム600を読み込み、実施例１の図２で示した方法と同一の手法により、プログラムの実行制御を行う。したがって、本実施例のリコンフィギャラブルプロセッサのプログラムは、異なるアレイサイズを持つリコンフィギャラブルプロセッサでも、同じプログラムを実行することができる。すなわち、プログラム互換性がある。

【００５４】

【発明の効果】

前述した実施例から明らかなように、本発明のプログラムは、ハードウェア（論理回路、プロセッサ）に与えるプログラム内に実行すべき演算と、その演算を実行するための依存関係（制約条件）を明示的に記述する。ハードウェアには、このプログラム内に記述された依存関係を基に実行順序を決定し実行する機構を設ける。これにより、スーパースカラのように依存関係を調査する専用ハードウェアを必要としないため、非常に少ないハードウェア量で済むと共に、VLIEのようにコンパイルの段階でスケジューリングを行うものではないので、異なるプロセッサ間でプログラムの互換性を保つことができる。

【００５５】

また、異なるサイズのリコンフィギャラブルプロセッサ上で、同一のプログラムを効率よく実行することができる。

【図面の簡単な説明】

【図１】

本発明の第１の実施例を示す図であり、プログラムと、そのプログラムを実行する論理回路の構成を示す図。

【図２】

図１のプログラムをデータフローグラフを用いて表現したプログラム記述と論理回路内の制御回路の構成とを示す図。

【図３】

本発明の第２の実施例を示す図であり、プログラムと、そのプログラムを実行するプロセッサの構成を示す図。

【図４】

本発明の第３の実施例を示す図であり、リコンフィギャラブルプロセッサを構成する演算回路アレイを示す図。

【図５】

図４の演算回路アレイを構成する演算回路セルの内部構造を示す図。

【図６】

図４の演算回路アレイからなるリコンフィギャラブルプロセッサを示す図。

【図７】

図６のリコンフィギャラブルプロセッサへ与えるプログラムの構造を示す図。

【図８】

図６の演算回路アレイに対するプログラムの構造を示す図。

【図９】

図２の実行演算選択部OSの処理内容の概略を示す図。

【図１０】

図３のディスパッチャDPTの処理内容の概略を示す図。

【図１１】

図２の演算管理部OMに格納される内容を示す図。

【図１２】

図２のデータ管理部DMに格納される内容を示す図。

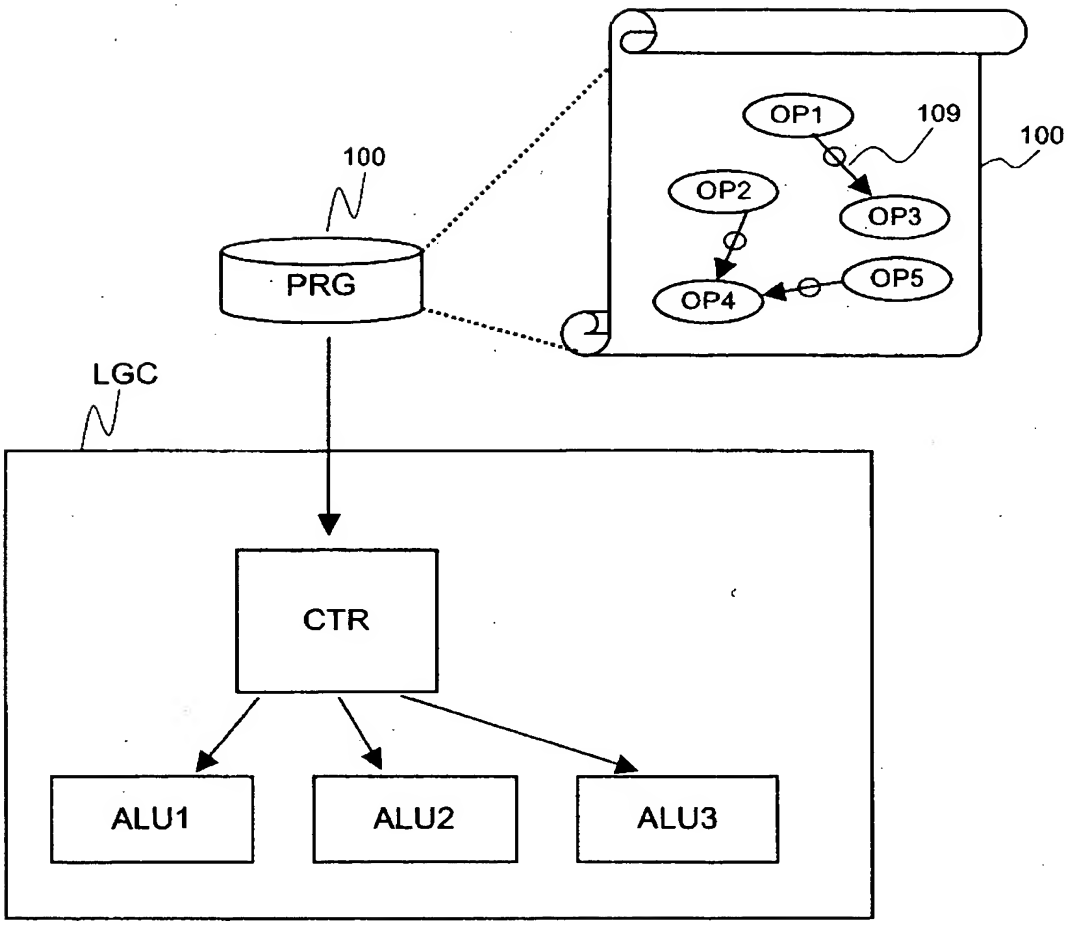
【符号の説明】

100…プログラム(PRG)、109…実行順序制約、122…出力データ（Out-data1,Out-data2）、123…入力データ（In-Data1～In-Data3）、200…プログラム、204…プロセッサ、209…実行順序制約、300…演算回路アレイ、302…データバス、303…コンフィグレーションバス、403…加算回路（ADD）、404…NAND回路、405…NOR回路、500…リコンフィギャラブルプロセッサ、501…接続素子、600…リコンフィギャラブルプロセッサ用プログラム、ALU1～ALU3…演算回路、ALUC…演算回路セル、CFG_MEM…コンフィグレーションメモリ、CFG_CTR…コンフィグレーション制御回路、CTR…制御回路、DM…データ管理部、DPT…ディスパッチャ、EIL…実行済み命令リスト、ALU-ARRAY_PRG1～ALU-ARRAY_PRG3…演算回路アレイに対するプログラム、ALUC_PRG1-1,ALUC_PRG1-2…演算回路セルに対するプログラム、INST1～INST5…命令、LGC…論理回路、MEM…メモリ、OM…演算管理部、OP,OP1～OP5…演算、OS…実行演算選択部、SEL…選択回路。

【書類名】図面

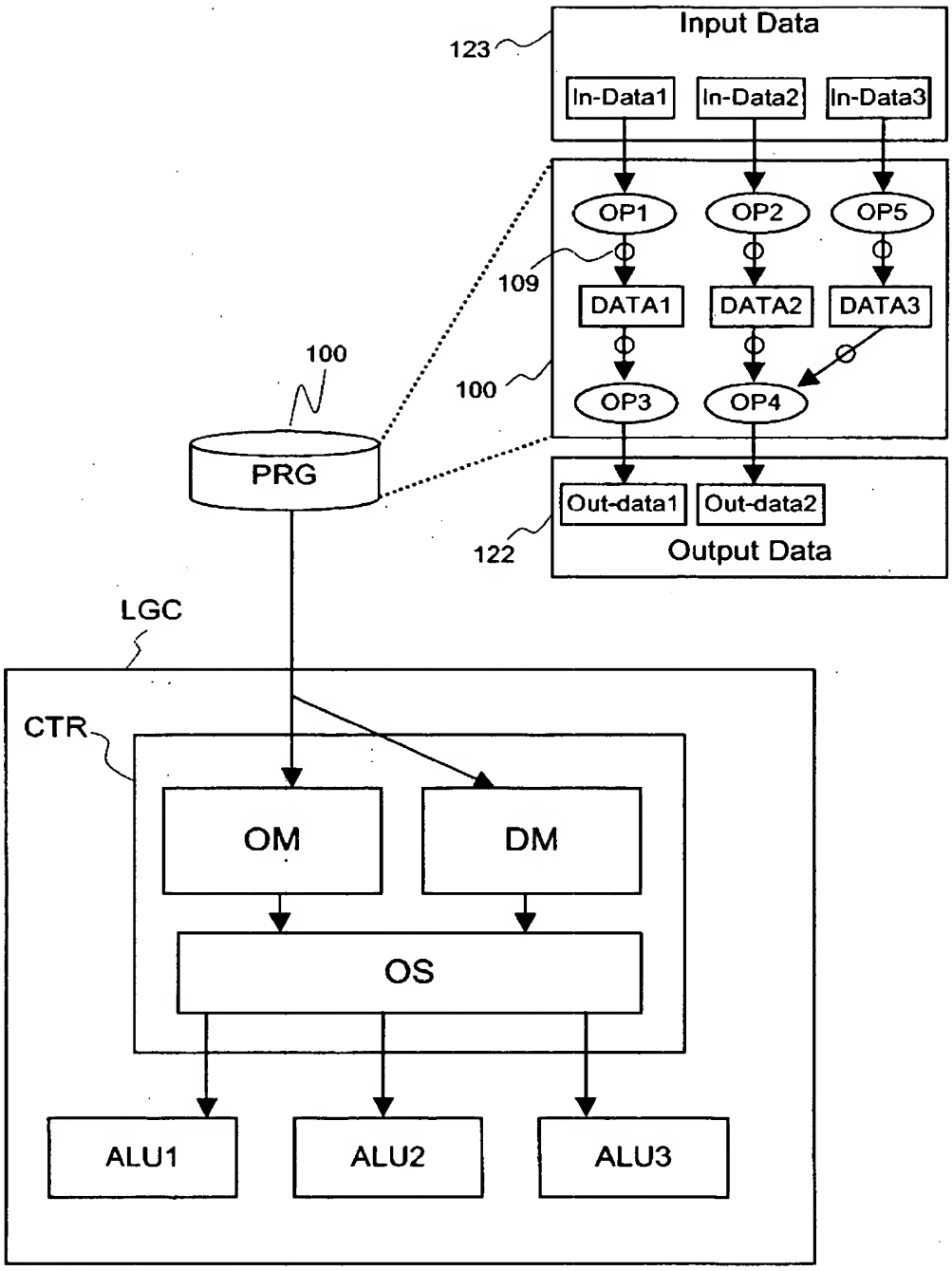
【図１】

図　1



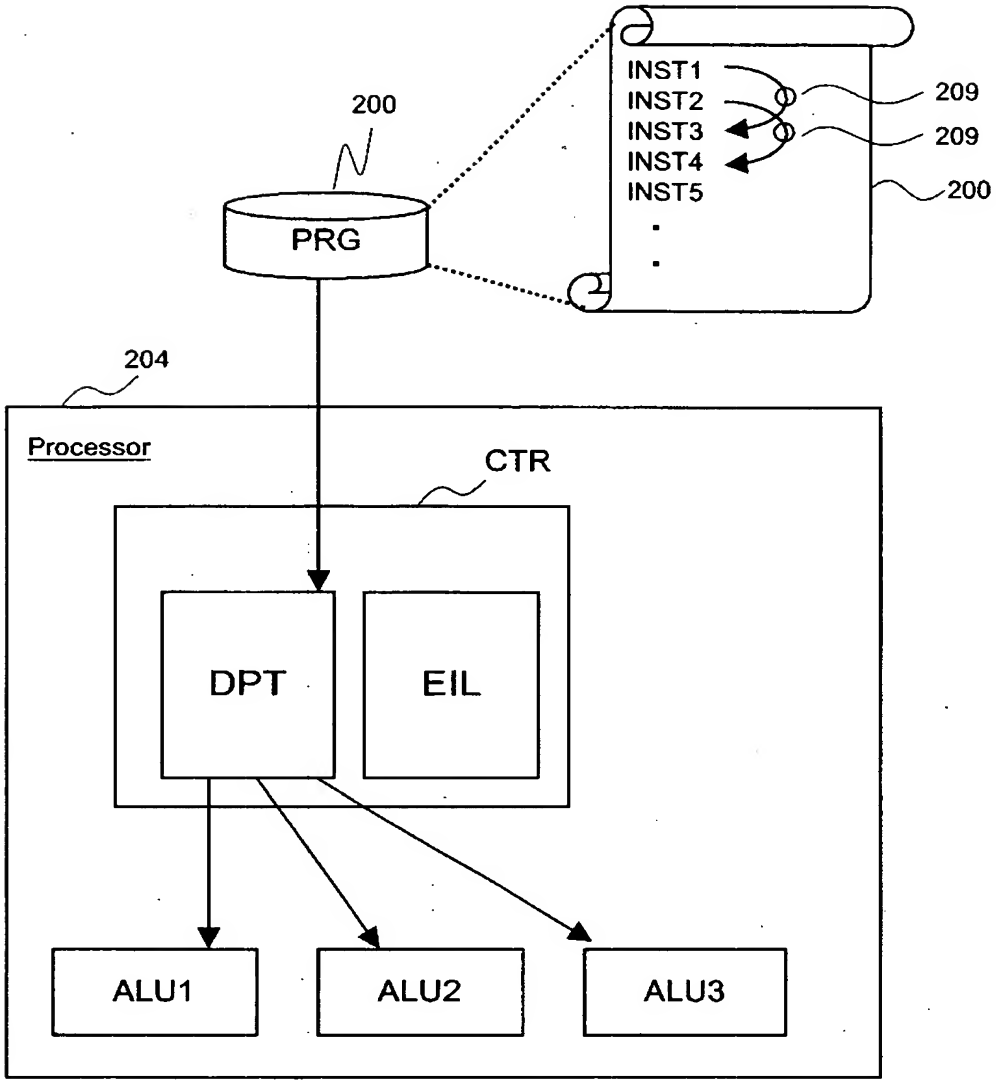
【図２】

図 2


Input Data
123
In-Data1
In-Data2
In-Data3
OP1
OP2
OP5
109
DATA1
DATA2
DATA3
100
100
OP3
OP4
PRG
Out-data1
Out-data2
122
Output Data
LGC
CTR
OM
DM
OS
ALU1
ALU2
ALU3

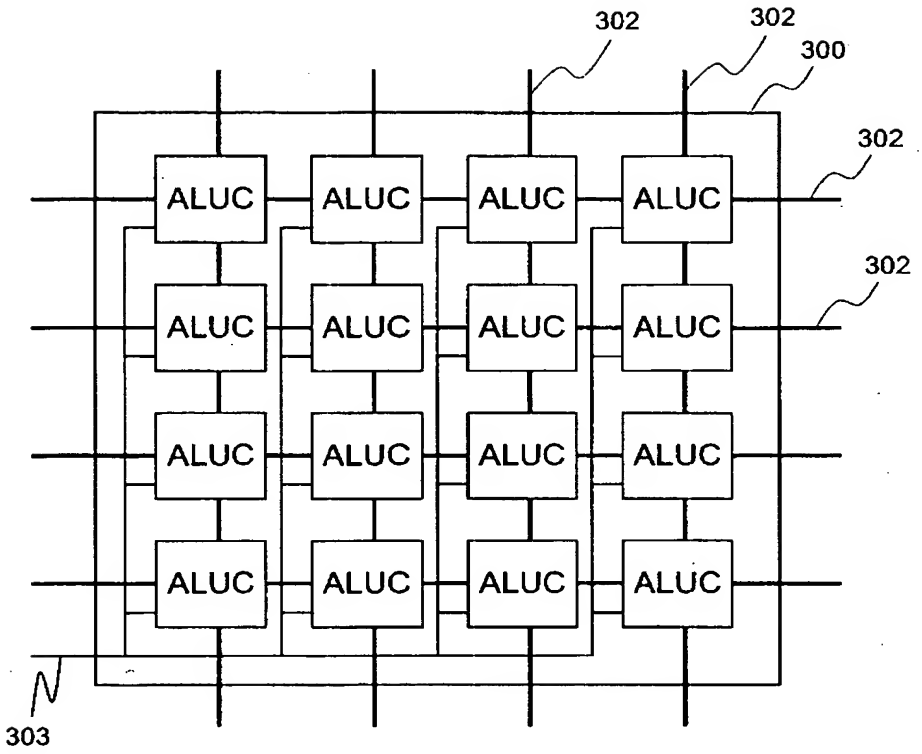
【図３】

図 3


200
PRG
INST1
INST2
INST3
INST4
INST5
209
209
200
204
Processor
CTR
DPT
EIL
ALU1
ALU2
ALU3

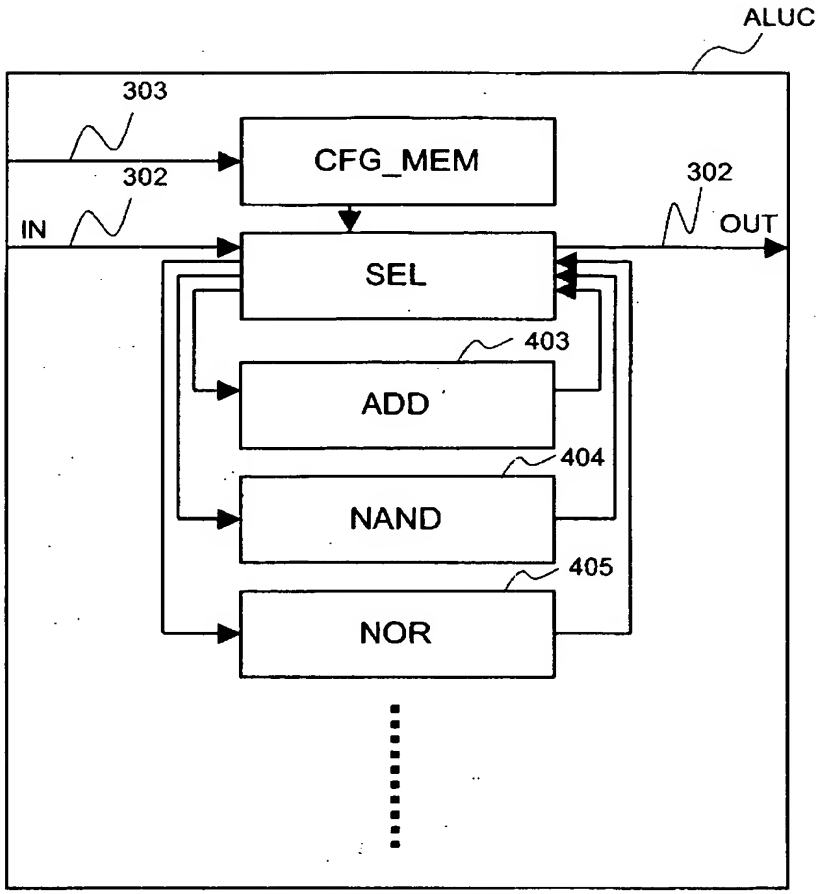
【図４】

図 4



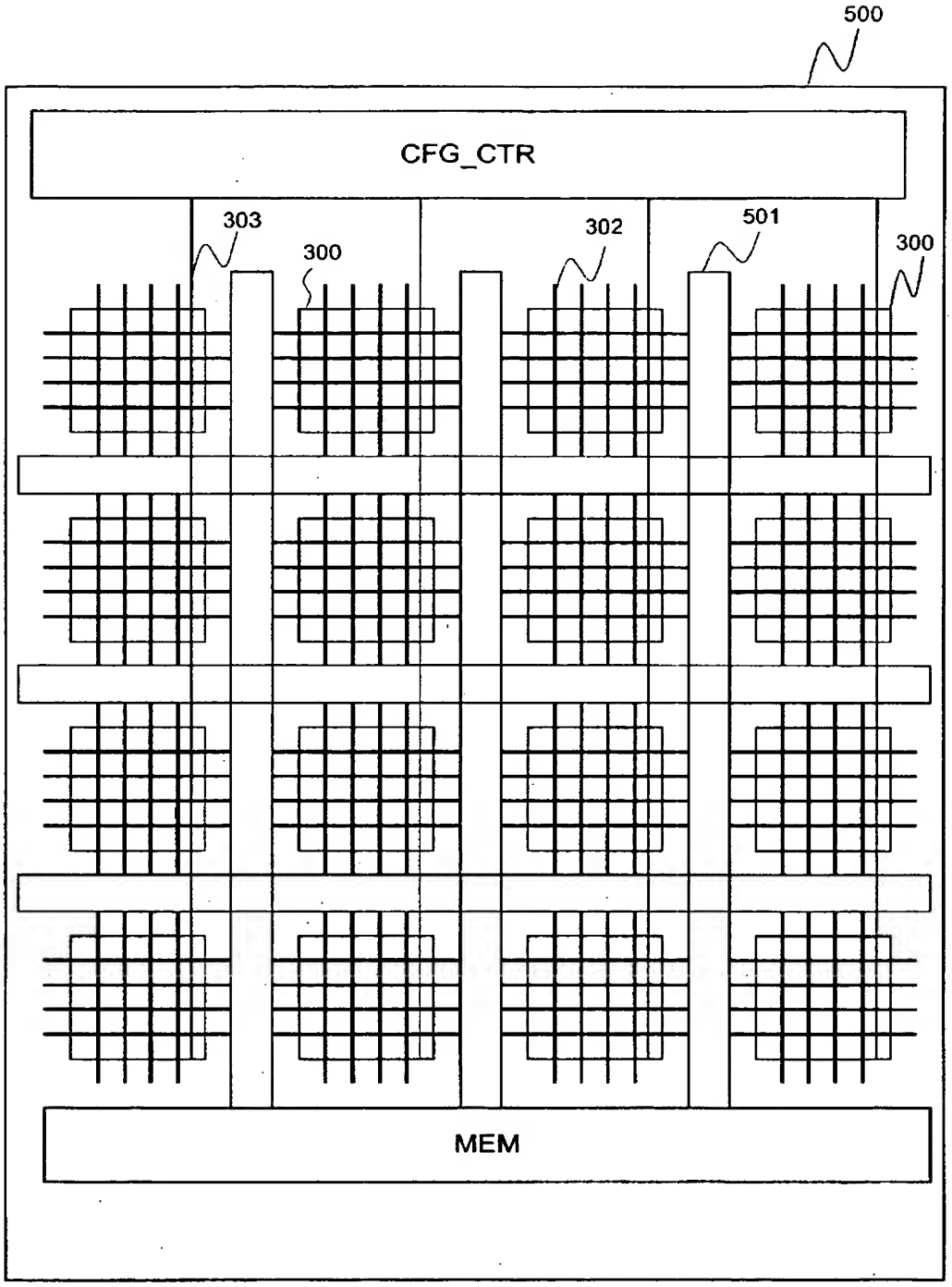
【図５】

図 5



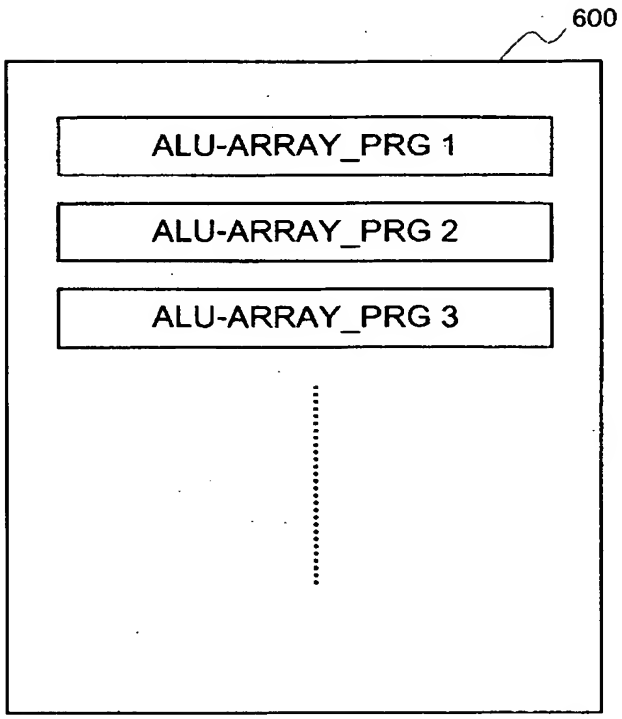
【図６】

図 6



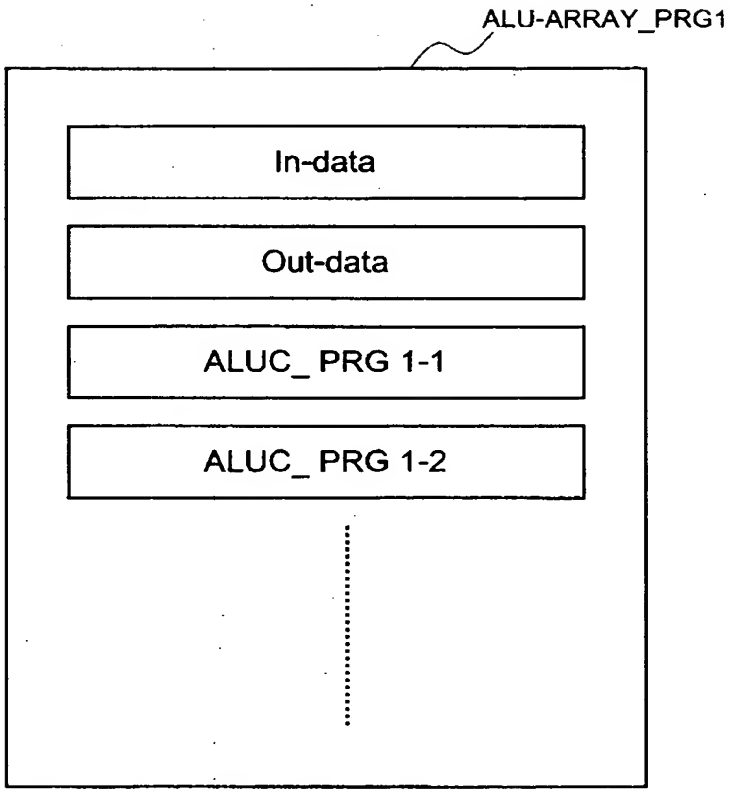
【図７】

図 7



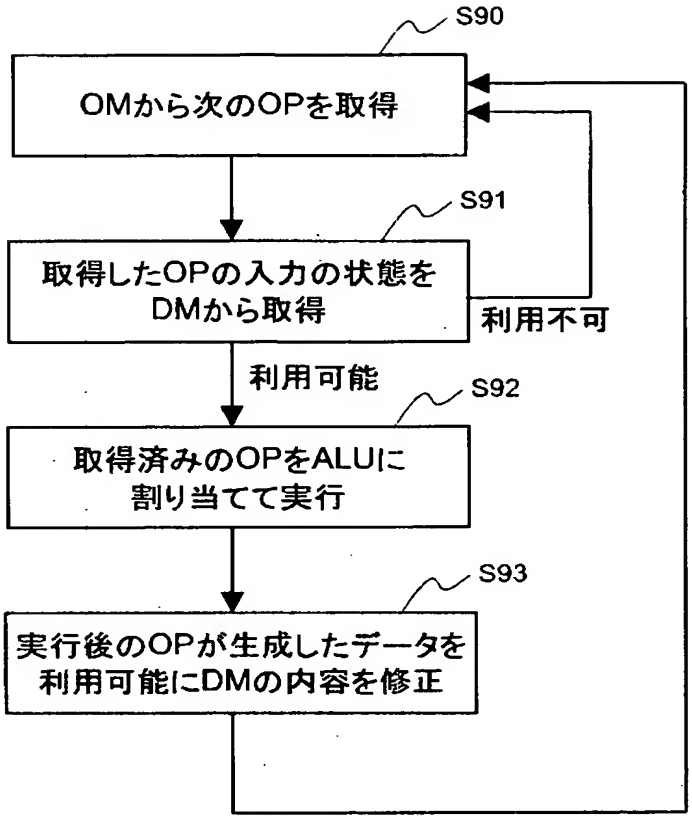
【図８】

図 8



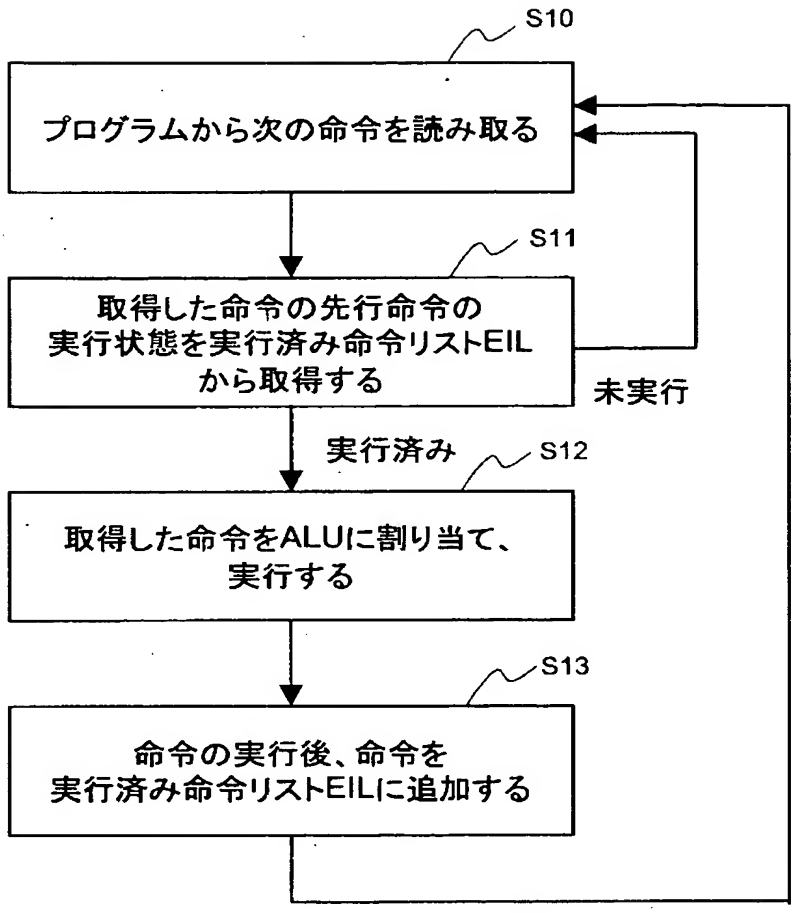
【図９】

図 ９


S90
OMから次のOPを取得
S91
取得したOPの入力の状態を
DMから取得
利用不可
利用可能
S92
取得済みのOPをALUに
割り当てて実行
S93
実行後のOPが生成したデータを
利用可能にDMの内容を修正

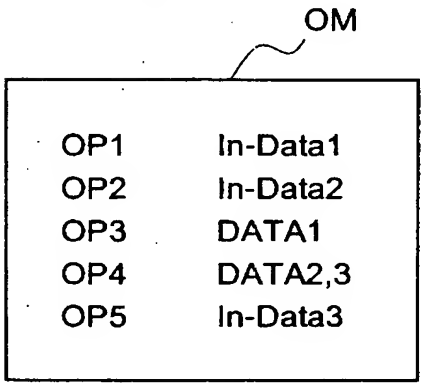
【図１０】

図 10



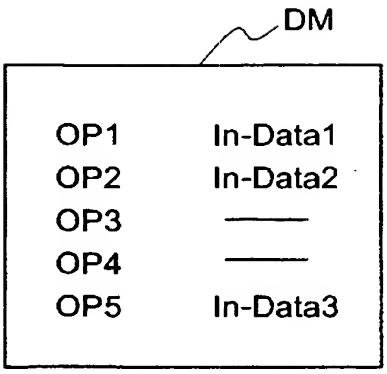
【図１１】

図 11



【図１２】

図 12



【書類名】要約書

【要約】

【課題】少ないハードウェア量で、異なるハードウェア間でのプログラム互換性を保ち、高い性能スケーラビリティを実現できるプログラムを提供する。

【解決手段】演算回路ALU1～ALU3と制御回路CTRから構成される論理回路LGC（ハードウェア）に与えるプログラム100内に、実行すべき演算OP1～OP5と、その演算を実行するための実行順序制約（依存関係）109を記述する。論理回路LGC内の制御回路CTRにより、読み込んだプログラム100に記述された依存関係109を基にして演算の実行順序を決定する。

【選択図】図１

特願２００３－１２８０８６

出　願　人　履　歴　情　報

識別番号　［０００００５１０８］

１．変更年月日　１９９０年　８月３１日
［変更理由］　新規登録
住　所　東京都千代田区神田駿河台４丁目６番地
氏　名　株式会社日立製作所